Chapter 12
Towards MDA Software Evolution

INTRODUCTION

This chapter discusses software evolution, challenges and strategic directions in the context of MDA.

Various authors agreed that it is difficult to define completely software and then, software evolution. Software is certainly more that bits stored in a file, it is an abstract idea that encompasses the concepts, algorithms embodied in the implementation as well as all its associated artifacts and processes. Research seems to confirm that computer software and process software have much in common. Osterweil (2003) assures that software processes are software too. In other paper (Osterweil, 2007), he suggests analyzing the nature of software and proposes to define taxonomies for exploring characteristics and approaches to the development, verification of qualities and software evolution. The exploration of these questions is an important current of software engineering research.

On the other hand, evolution is defined as a process of gradual change and development from fewer and simpler forms to higher, more complex, or better ones. In biology, evolution is related to develop over time often many generations, into forms that are better adapted to survive changes in their environment. Thus, evolution captures the notion of something improving and changes occur in species in successive generations, i.e. individuals get old and species evolve. Jazayeri (2005) analyzes the definition of software evolution. The concept of “specie” in software may be associated to meta-levels describing families (species) of software systems. These meta-levels or architectures are created as improvements to previous existing ones and describe evolved families of software systems.

Evolution must focus on “species” of software rather than individual software applications. Then, “Software, like people, get old“ (Parnas, 1994) and meta-levels or architectures, like species, evolve.

DOI: 10.4018/978-1-61520-649-0.ch012
Towards MDA Software Evolution

Starting from our understanding of software applications, their specification in terms of models (and metamodels) evolves to new generation of tools that have an improved structure and are based on new technologies.

Software evolution is multidimensional and is composed of different types of entities/concepts or artifacts that come from specifications, designs and architectures to source code, test cases and documentation. Each of them depends on other artifacts embodied in the implementation such as user interfaces, components, patterns and so on. The different ways and rates that these artifacts change, lead to unreliable software and cause many problems associated with software maintenance.

Software artifacts evolve at different rhythm and in different ways, for instance, an initial design is not updated to reflect the changes that are introduced in the code. Software evolution needs a software development framework that supports the consistency evolution of the different dimensions of the software.

In this light, an MDA-based approach can help to support consistently software evolution due to MDA can be viewed as an integration architectural framework that maintains consistency as the software evolves, i.e., the concept of multidimensional evolution is in the essence of MDA.

MDA can help to develop and support a common application framework for software evolution that raises issues such as common exchange formats, tool integration and interoperability. When the system evolves, MDA maintains the interrelation between software entities accommodating the evolution of higher level artifacts together with the code in a consistent way.

Next, we describe challenges in software evolution and the role of MDA to overcome or avoid the negative effect of software aging.

CHALLENGES ON MDA-BASED SOFTWARE EVOLUTION

(Mens, Wermelinger, Ducasse, Demeyer, Hirschfeld and Jazayeri, 2005) list 18 essential challenges in the software evolution that need to be addressed in the future. We include the main challenges that our MDA-based reverse engineering and software evolution could overcome:

• “To provide tools and techniques which preserve or even improve the quality characteristics of a software system, whatever its size and complexity”.
• “To develop and support a common application framework for doing joint software evolution research”.
• “Software evolution techniques should be raised to a higher level of abstraction, in order to accommodate not only evolution of programs, but also evolution of higher-level artifacts”.
• “To achieve co-evolution between different types of software artifacts or different representation of them”.
• “In order to become accepted as practical tools for software developers, formal methods need to embrace change and evolution as an essential fact of life”.
• “Software evolution must provide more and better support for multi-language systems.”
(Mens, Wermelinger, Ducasse, Demeyer, Hirschfeld and Jazayeri, 2005)

A challenge on software evolution is the necessity to achieve co-evolution between different types of software artifacts or different representations of them. MDA allows us to develop and relate all dif-
Related Content

An Integrated Infrastructure Using Process Mining Techniques for Software Process Verification
www.igi-global.com/chapter/an-integrated-infrastructure-using-process-mining-techniques-for-software-process-verification/192933?camid=4v1a

Sequential Test Set Compaction in LFSR Reseeding
www.igi-global.com/chapter/sequential-test-set-compaction-lfsr/51415?camid=4v1a

Wavelet Transform Algorithms
www.igi-global.com/chapter/wavelet-transform-algorithms/202840?camid=4v1a

Code Clone Detection and Analysis in Open Source Applications
www.igi-global.com/chapter/code-clone-detection-and-analysis-in-open-source-applications/192915?camid=4v1a